МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Інститут інженерії та інформаційних технологій
(повне найменування інституту, назва факультету)

Кафедра комп'ютерної інженерії та електромеханіки
(повна назва кафедри)

ДИПЛОМНА БАКАЛАВРСЬКА РОБОТА

на тему

Розроблення JavaScript програми для контролювання робота Arduino

Виконав: студент групиБКІ-18

спеціальності 123 «Комп'ютерна інженерія»
(шифр і назва спеціальності)

Владислав СТАРИНЕЦЬ
(прізвище та ініціали)

Керівник д.т.н., проф. Володимир ОСИПЕНКО
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Київ 2022

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Інститут інженерії та інформаційних технологій
Кафедра комп'ютерної інженерії та електромеханіки
Спеціальність 123 «Комп'ютерні інженерія»
Освітня програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ

Завідувач кафедри КІЕМ

_____проф. Борис Злотенко

"_____" _____2022 року

З А В Д А Н Н Я

НА ДИПЛОМНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Старинцю Владиславу Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема дипломної бакалаврської роботи **Розроблення JavaScript програми для контролювання робота Arduino**

Науковий керівник роботи Осипенко Володимир Васильович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
д.т.н., професор

затверджені наказом вищого навчального закладу від 15.03.2022 № 75-уч.

2. Строк подання студентом роботи 9 червня 2022 року

3. Вихідні дані до дипломної бакалаврської роботи:приладипрограмування робота на Arduino за допомогою трьох бібліотек з використанням JavaScript; навчальна та методична література; державні стандарти.

4. Зміст дипломної бакалаврської роботи (перелік питань, які потрібно розробити): 1. Провести аналітичний огляд бібліотек для керування робота. 2. Описати конструкція робота на Arduino. 3.Описати роботу бібліотек.

5.Консультанти розділів дипломної магістерської роботи

Розділ	Ім’я, прізвище та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Вступ	д.т.н., проф. Володимир Осипенко		
Розділ 1	д.т.н., проф. Володимир Осипенко		
Розділ 2	д.т.н., проф. Володимир Осипенко		
Розділ 3	д.т.н., проф. Володимир Осипенко		
Висновки	д.т.н., проф. Володимир Осипенко		

6.Дата видачі завдання14 березня 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної бакалаврської роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	15.04.2022	
2	Розділ 1. Аналітичний огляд методів та засобів вимірювання твердості матеріалів	20.04.2022	
3	Розділ 2. Конструкція приладу та підбір складових	25.04.2022	
4	Розділ 3. Опис роботи приладу	06.05.2022	
6	Висновки	10.05.2022	
7	Оформлення дипломної бакалаврської роботи (чистовий варіант)	06.06.2022	
8	Здача дипломної бакалаврської роботи на кафедру для рецензування (за 14 днів до захисту)	10.06.2022	
9	Перевірка дипломної бакалаврської роботи на наявність ознак плагіату (за 10 днів до захисту)		
10	Подання дипломної бакалаврської роботи на затвердження завідувачу кафедри (за 7 днів до захисту)		

Студент _____ Старинець В.В.
(підпис) (прізвище та ініціали)

Науковий керівник роботи _____ Осипенко В.В.
(підпис) (прізвище та ініціали)

Рецензент _____ (прізвище та ініціали)
(підпис)

АНОТАЦІЯ

Старинець В. В. Розроблення JavaScript програми для контролювання робота на Arduino. – Рукопис.

Дипломна бакалаврська робота за спеціальністю 123 Комп'ютерна інженерія, освітньою програмою «Комп'ютерні системи та мережі». – Київський національний університет технологій та дизайну, Київ, 2022 рік.

Дипломну бакалаврську роботу присвячено порівнянню JavaScript бібліотек для керуванням робота на мікроконтролері Arduino.

У роботі запропоновано використання бібліотек JavaScript для керування роботом. У якості робота було використано мікроконтролер Arduino, для якого потрібно мінімальна кількість матеріалів, щоб його зібрати. Для програмування робота було запропоновано використання IDE – ArduinoIDE, в якому можна встановити додаткові інструменти, щоб підключитися до робота. Необхідним інструментом у виконанні дипломної роботи, був запуск серверу на NodeJS, за допомогою якого можна було встановлювати бібліотеки. Для того, щоб підключитися до мікроконтролера, використовувався USB-кабель, який слід під'єднати до комп'ютера.

Досліджено принципи роботи бібліотек, продемонстровано життєвий цикл їх методів, показано роботу при під'єднанні, від'єднанні від джерела живлення мікроконтролера.

Ключові слова: JavaScript, NodeJS, Arduino, Babbler_h, мікроконтролер, Johnny-five, Serialport.

ABSTRACT

Starynets V. V. Developing JavaScript program to control Arduino robot. – Manuscript.

Bachelor's thesis in specialty 123 «Computer Engeneering», educational program «Computer systems and networks». – Kyiv National University of Technology and Design, Kyiv, 2022.

Thesis is developer to compare JavaScript libraries to control robot on Arduino microcontroller.

This paper proposes the use of JavaScript libraries to control the robot.As the robot was used Arduino microcontroller, which needs minimum stuff to construct it. To program the robot was proposed the use of Arduino IDE in which you can install additional tools to connect to the robot. Needed tools in thesis was start own NodeJS server to install libraries to program. USB-cable was used to connect microcontroller to computer.

It was possible to investigate the principles of libraries work, there were demonstrated lifecycles of libraries methods, also demonstrated work with connection\disconnection from source of power.

Keywords: JavaScript, NodeJS, Arduino, Babbler_h, microcontroller, Johnny-five, Serialport.

ЗМІСТ

ВСТУП

Актуальність роботи.

Сьогоднішні реалії дуже важко уявити без технічних пристроїв. Завдяки розвитку технічної сфери, людство має безліч гаджетів, контролерів, машин і т.д. У звязку з розвитком терміну makermovement(термін для визначення розробників, чия діяльність пов’язана з технікою) звявилася велика кількість розробників программного забезпечення, які хочуть створити свого робота та керувати ним.

На сьогоднішньому ринку є дуже багато мікроконтролерів, за допомогою яких, можна створити власну систему керування пристрою. Одним із таких чудо-пристроїв є мікроконтролер Arduino. Все що потрібно для керування, це сам контролер Arduino,USB-кабель, який потрібно підключити до комп’ютера, та IDE(IntegratedDevelopmentEnvironment – інтегроване середивище розробки), в якому можна запрограмувати робота.

Можна створити, як звичайного робота, який з певним інтервалом буде вмикати\вимикати лампочку, так і систему, яка буде слідкувати за температурою в кімнаті за допомогою датчиків, відкривати вікно, коли температура висока, або закривати його, коли температура опустилася. Завдяки таким ідеям, мікроконтролери стали досить популярними і налічують дуже багато видів.

Але, на жаль, більшість контролерів написані на мові програмування Sketch – мові програмування на основі C, з якою багато розробників почувають себе некомфортно.

У звязку з вище перерахованою проблемою, я б хотів запропонувати, як можна контролювати робота на Arduino, за допомогою такої мови програмування, як JavaScript.

Метою роботиє вибір JavaScriptбібліотеки для контролювання робота на Arduino, серед широкого спектру інструментів.

Для досягнення поставленої мети у роботі було вирішено такі **задачі**:

- проведено аналітичний огляд сучасних бібліотек для контролювання робота на Arduino;
- запропоноване технічне рішення по вибору найкращої технології для роботи з мікроконтролером;
- розробленоряд програм для робота, за допомогою яких можна керувати мікроконтролером Arduino.

Об'єкт дослідження –процес керування робота на Arduino.

Предмет дослідження –керування роботом на Arduino, за рахунок бібліотек мови програмування JavaScript.

Методи досліджень. Базою дослідження стали основні положення робототехніки , машинобудування та програмування.

Інформаційною базою досліджень євідкриті джерел Internet, наукові, технічні статті, документації.

Практичне значення отриманих результатів.

У роботі запропоновано технічне рішення контролювання робота на Arduino, за допомогою бібліотеки Johnny-five, завдяки її простоті, надійності в користуванні, а також завдяки кросс-платформенності бібліотеки, за рахунок чого одний і той же код, буде корректно працювати на різних пристроях.

Структура та обсяг роботи.Дипломна робота бакалавра складається зі вступу, 3 розділів та висновків по них, загальних висновків, списку використаних джерел та додатків. Основний текст роботи викладений на 50 сторінках, містить 37 рисунок, список джерел з 27 найменувань. Загальний обсяг роботи, враховуючи додаток, складає 55аркуша.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ ПРОГРАМУВАННЯ РОБОТА ARDUINO НА NODEJS

1.1. Теоретичні відомості про NodeJs

NodeJS – JavaScript фреймворк, побудований на основі двигуна ChromeV8, який використовується для розроблення інтенсивних веб-додатків Входу\Виходу, таких як SPA(SinglePageApplication), сайти потокового відео і т.д.

NodeJS використовує асинхронне програмування, однопоточність, не блокуючу модель викликів і являється дуже ефективним фреймворком, який використовує пам’ять.

Потік виконання коду оснований на циклу подій.

Цикл подій – безкінечний цикл, в якому двигунJavaScript постійно очікує нові задачі, виконує їх і знову чекає на появу задач.

Алгорит циклу подій виглядає наступним чином(Рис.1):

1. Поки задачі існують – виконує їх, починаючи з найдавнішої.
2. Якщо задач немає, двигун JavaScript бездіє, до того моменту, поки не появиться нова задача і потім виконується 1 пункт.

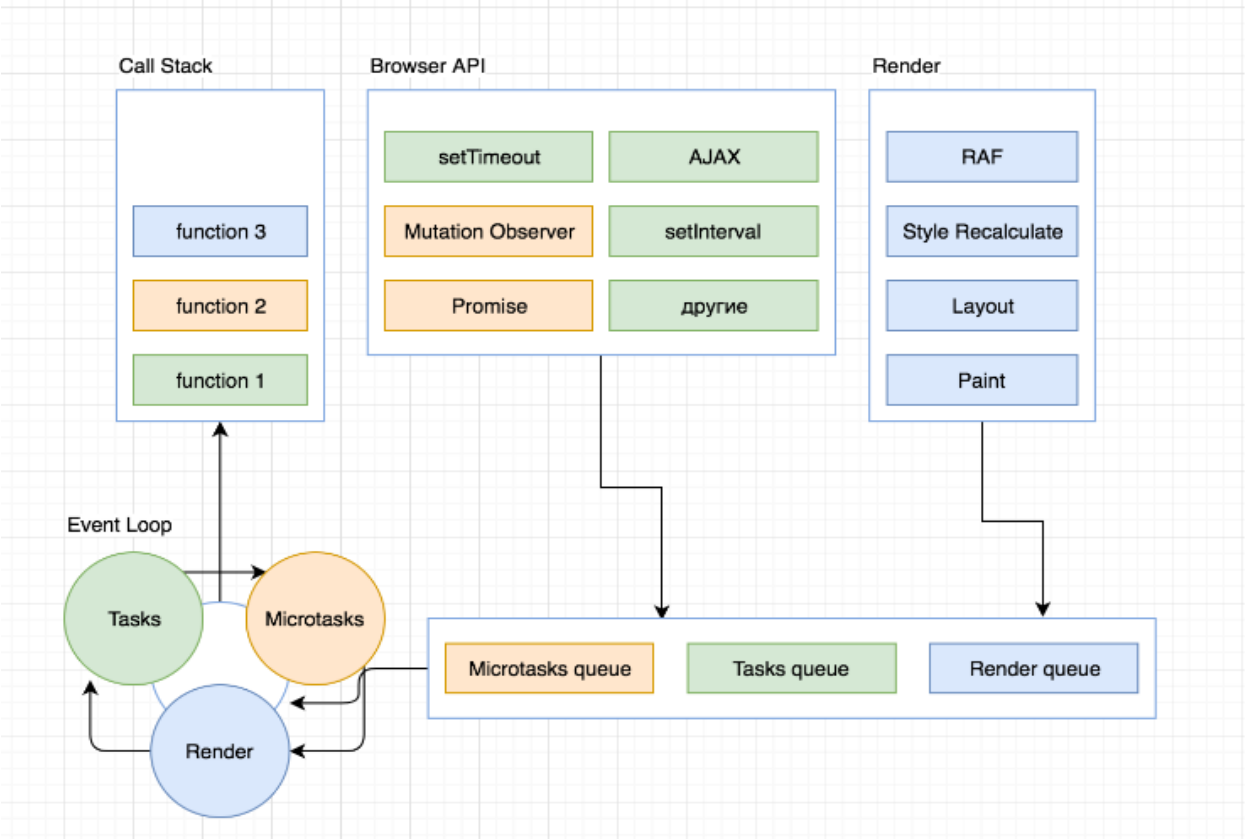

Рис1. Робота EventLoop

Двигун JavaScript більшість часу нічого не робить і працює лише тоді, коли потрібно обробити нову подію. Задачі попадає на виконання – двигун їх виконує – і знову очікує на нові задачі (під час очікування майже не загружається процесор на комп'ютері). Якщо задача попадає на виконання, коли двигун уже виконує задачу, вона стає в чергу. Черги, які формують такі задачі, називаються чергами макрозадач.

Задачі в чергах виконуються по правилу "перший зайшов – перший вийшов".

Незважаючи на макрозадачі, також присутні мікрозадачі. Мікрозадачі виникають лише під час виконання коду. Наприклад, ми дали запит знайти файл у файловій системі компютера і через деякий час, коли все буде опрацьовано, нам повернеться Promise, який являється мікрозадачею.

Якщо одночасно виконається мікрозадача, та макрозадача, то двигун JavaScript повернее першою виконану мікрозадача, так як вона йде більше в приоритеті, а потім повернеться виконана макрозадача. Всі мікрозадачі завершують своє виконання до обробки яких-небуть подій, або до переходу до іншої макрозадачі.

Саме завдяки циклові подій, однопоточний NodeJSне блокується.

Додаток, написаний на NodeJS, може одночасно обробляти багато з'єднань. На кожне з'єднання викликається функція зворотнього виклику, але якщо додаток перестає отримувати з'єднання, він засинає.

Простим завданням для веб-додатку може бути прочитати файл на сервері та повернути результат до клієнта, ось як NodeJS обробляє даний запит:

- Клієнт надсилає запит на сервер.
- Сервер отримує запит і викликає функція зворотнього виклику, яка шукає файл у файловій системі.
- Сервер готовий обробити новий запит.
- Коли файл буде відкрито і прочитано, сервер поверне результат до клієнту.

Також за допомогою NodeJS можна:

- Створювати, відкривати, редагувати, удаляти файли на сервері.
- Динамічно генерувати контент для веб-сторінок.
- Збирати дані з форм.

1.2.Запуск сервера на NodeJS

Для того, щоб запустити сервер на NodeJS, нам потрібно встановити NodeJS глобально на свій комп’ютер. Завантажити .exe файлми можемо на офіційному сайті NodeJS.org. Далі потрібно запустити .exe файл, після чого зв’явиться встановлювач NodeJS(Рис2).

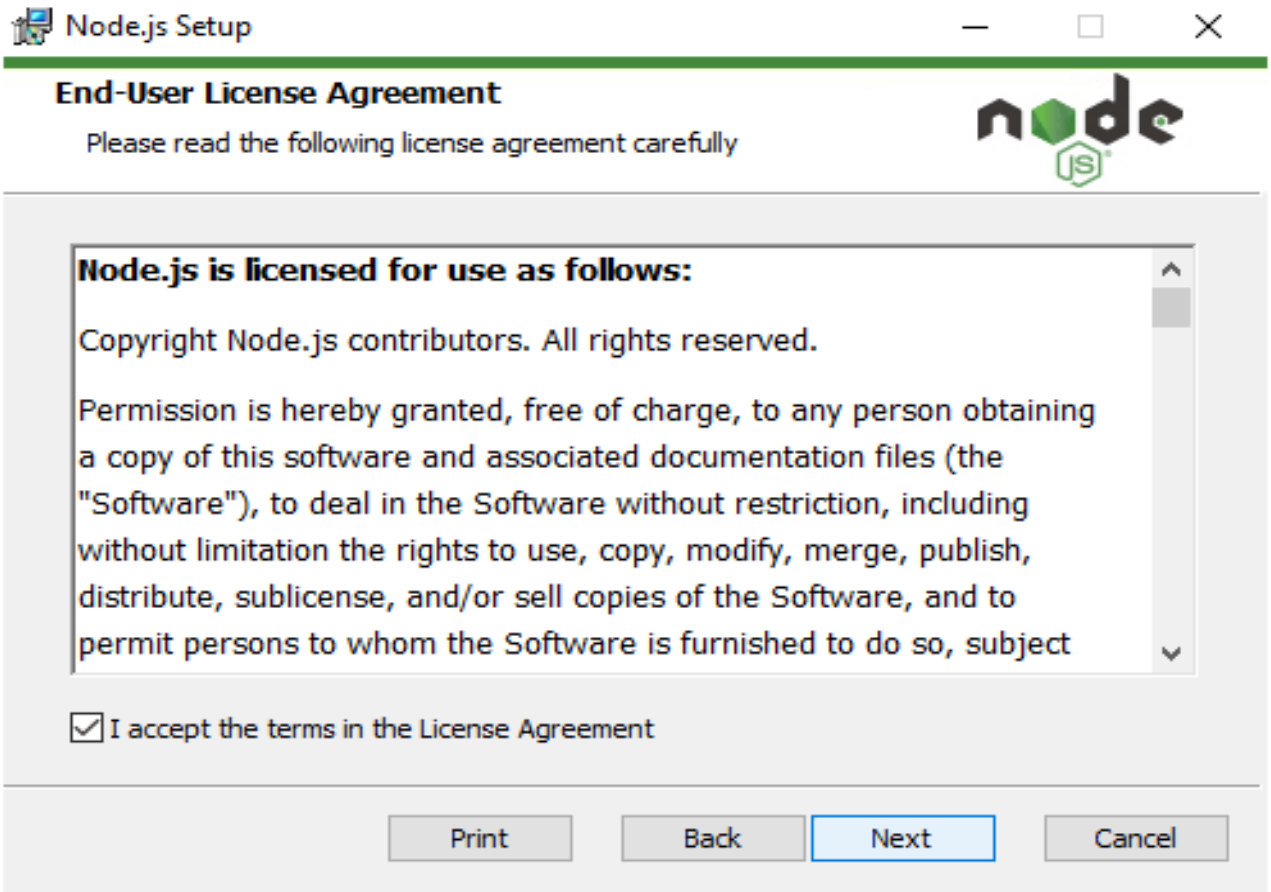

Рис 2. Встановлювач NodeJS.

Пройшовши інструкцію по встановленню, потрібно перезавантажити комп’ютер. Для того, щоб перевірити чи все встановилось правильно, потрібно відкрити CommandPrompt і прописати команду `**node -v**`. В консолі буде відображена NodeJS версія, яка була завантажена.

Для того, щоб запустити сервер, потрібно створити файл з розширенням *.js і написати наступний код(Рис.3):

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Рис.3 Запуск сервера на NodeJS

Далі потрібно зберегти і назвати файл, наприклад **server.js**. Для того, щоб запустити сервер, в Command Prompt потрібно написати команду **nodeserver.js**, після чого в консолі ми побачимо **Serverrunningathttp://127.0.0.1:3000.**

Після того, як сервер запущено, можна використати одну з нижче вказаних бібліотек, для програмування робота Arduino.

1.3 Бібліотека Serialport

Serialport – це JavaScript бібліотека, яка з’єднує послідовні порти, які працюють в NodeJStaElectron.

Переваги використання Serialport:

- Один з найпростіших способів для комунікації з Arduino.
- Легка для прочитання та розуміння документація.
- На просторах Інтернету є дуже багато статей про Serialport.

Недоліки:

- Не можна робити складні вичислення.
- Код в Arduinopотрібно писати на C++.

1.4 Бібліотека Babbler_h

Babbler_h – це клієнтська бібліотека, для комунікації з Arduino пристроями.

Переваги:

- Бібліотека самостійно відповідає за підключення, вся логіка інкапсульована: слідкує за розривами з'єднання, повідомляє про зміну статусу, перепідключається заново.
- Команди надсилаються одна за одною, не блокується потік виконання коду.
- Для того, щоб пристрій був підключений потрібно, щоб було виконано дві умови: відкритий канал звязку, пристрій надіслав статус: **OK**на команду ping.
- Бібліотека дуже добре налаштована для некорректної роботи робота: робот може надіслати некорректну команду, не вчасно відправляти запити. В цьому випадку бібліотека буде або ігнорувани некорректну роботу робота, або повідомить користувача, що робот не виконав команду.
- Бібліотека завжди повідомить користувача, про завершення виконання команди.

Недоліки:

- Робот отримує і відправляє команди лише в JSONформаті.
- Пристрій повинен відповідати на команди не більше 5 секунд, в іншому випадку буде помилка виконання.
- Прошивка робота обовязково повинна містити команду ping.

Архітектурно бібліотека складається з 3 частин:

- Модуль каналів зв’язку (включає в себе роботу з послідовним каналом, вайфай, блютуз): встановлення та обслуговування з’єднання, відправлення відповіді.
- Модуль команд : реєстарція функції у вигляді команди, пошук команди по назві, виконання команди.
- Допоміжні протоколи: формати для отримання команд, та упакування відповіді (команда і відповіді у форматі JSON)

Робот повинен приймати команди в форматі JSON. Пакет даних – рядок, який повинен містити команду або відповідь (Рис.4).

```
{"cmd": "help", "id": "34", "reply": "help ping ledon ledoff"}
```

Рис.4 формат JSON

Де:

cmd – назва команди

id – клієнтський ідентифікатор команди

params – параметри команди

1.5 Бібліотека Johnny-five

Johnny-five – це JavaScript платформа робототехніки, яка створює з'єднання між Arduinota комп'ютером.

Переваги:

- Бібліотека дозволяє працювати не тільки з проектами Arduino, а й з платформами, які мають Вхід\Вихід.
- Дуже проста у використанні.
- Бібліотека має велику аудиторію, тим самим код дуже легко підтримувати.

Недоліки:

- Потрібно завантажувати додатковий протокол Firmata

Для початку потрібно підключити Arduinoза допомогою USB-кабелю до комп'ютера. Також для роботи, потрібно завантажити Arduino IDE, яка необхідна лише для початкового налаштування. В налаштуваннях Arduino IDE потрібно перейти в Інструменти >Порти (Рис.5), та впевнетися, що необхідна плата Arduino під'єднана.

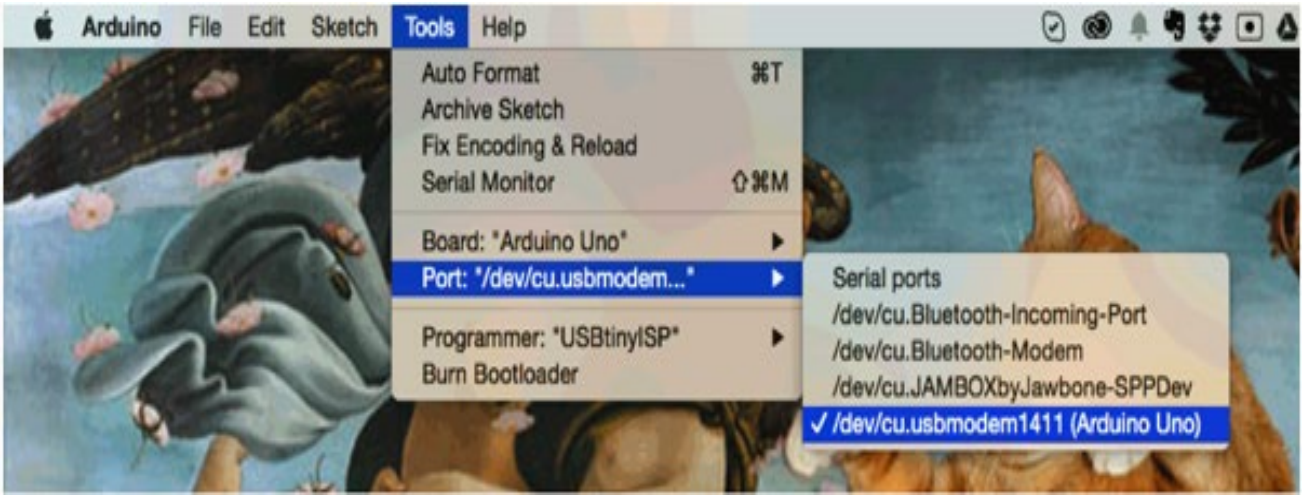

Рис.5 Приклад підключення порту

Johnny-Five взаємодіє із Arduino за допомогою протоколу Firmata, тому потрібно встановити StandardFirmata. Для цього в IDEпотрібно відкрити Файл > Приклади >Firmata>StandartFirmata (рис.6), натиснути кнопку upload, дочекатися завантаження протоколу, далі для роботи IDE не потрібна.

Рис. 6 Приклад вставлення StandardFirmata

Висновки до розділу 1

1. Описано області застосування фреймворку NodeJS.
2. Розглянуто сильні сторони програмування на NodeJS, та як даний фреймворк працює асинхронно.
3. Розглянуто, як потрібно встановити NodeJS та запустити сервер.
4. Наведено основні бібліотеки, за допомогою яких, можна контролювати робота на Arduino.
5. Описано особливості роботи бібліотеки.
6. Наведено переваги та недоліки бібліотек.

Під час виконання аналізу бібліотек для програмування робота на Arduino, було розгялнуто основні характеристики і можливості бібліотек. Для деяких бібліотек дотримуватися вказаних вимог, щоб контролювати робота.

РОЗДІЛ 2. КОНСТРУКЦІЯ ПРИЛАДУ ARDUINO

2.1. Конструкція приладуArduino

Для складання мікроконтроллера було використано такі деталі (Рис 7):

- Arduino (Genuino) Uno
- 1 RGB кабель
- 1 прототипна плата
- 4 дротяні перемички з двома штировими кінцями (червона, чорна, зелена, синя)
- 1 резистор

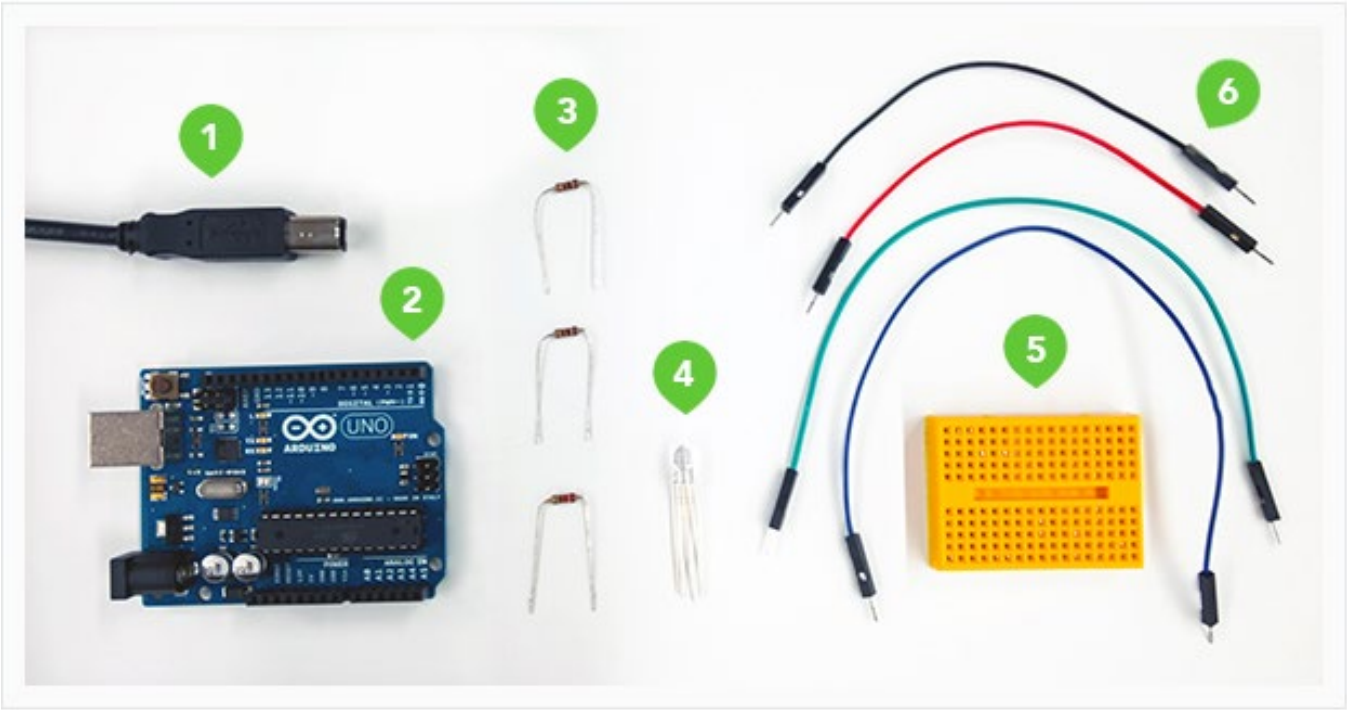

Рис 7. Деталі для складання Arduino

На рис. 8 показано схема складеного Arduino

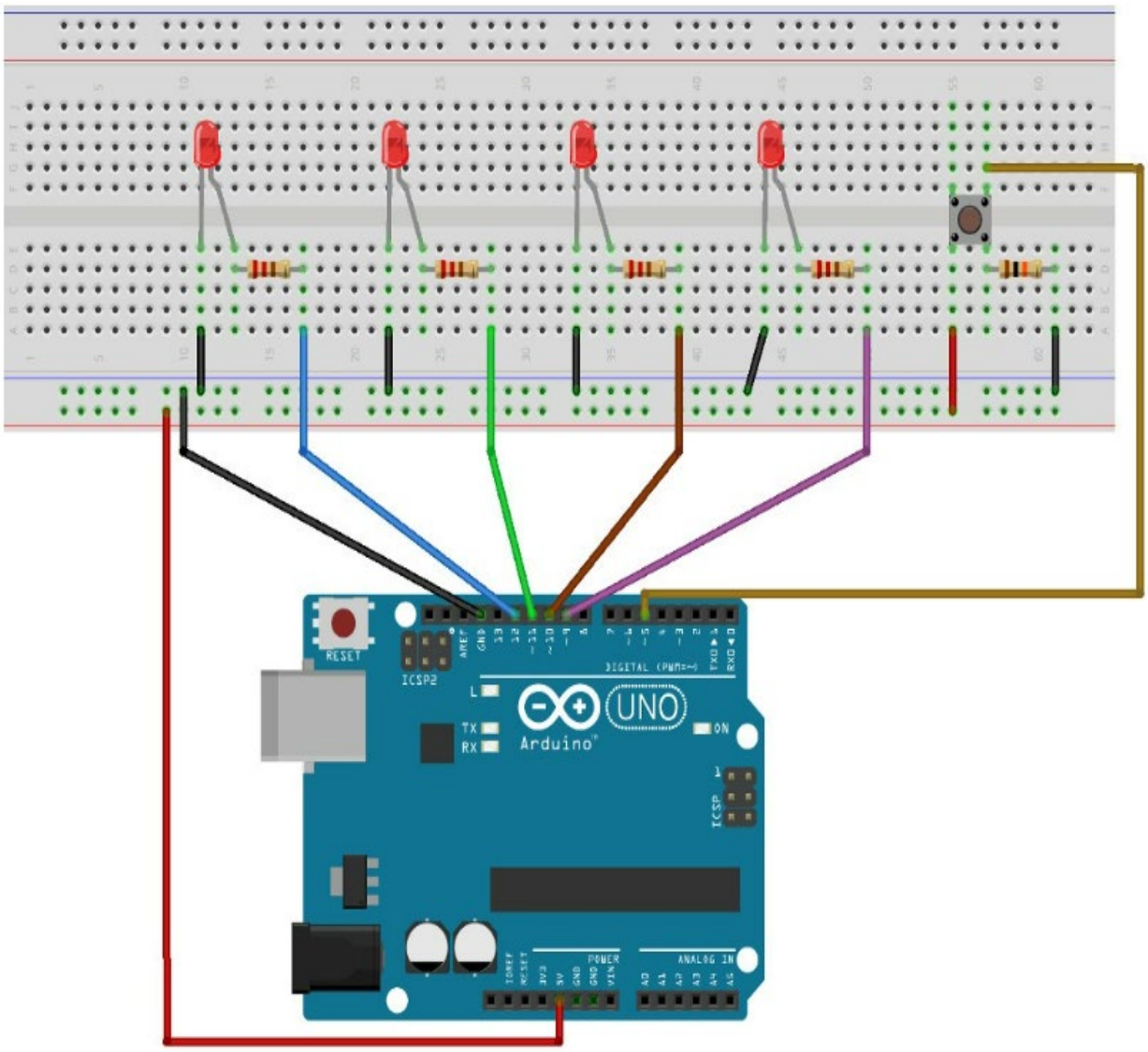

Рис.8 Схема складеного Arduino

Arduino – це пристрій на основі мікроконтролера ATmega328 (datasheet). Він вмістив у собі все необхідне для комфортної роботи з мікроконтролером: цифрових виходів\входів - 14, аналогових входів - 6, кварцовий резонатор на 16 МГц, роз'єм для внутрішньо схемного програмування, роз'єм живлення, роз'єм USB і кнопку скидання. Для того, щоб розпочати роботу з пристроєм просто потрібно подати живлення від AC/DC-адаптера або батарейки, або використати USB кабель, який потрібно підключити до комп’ютера.

Характеристики (Рис.9)

Микроконтроллер	ATmega328
Рабочее напряжение	5В
Напряжение питания (рекомендуемое)	7-12В
Напряжение питания (предельное)	6-20В
Цифровые входы/выходы	14 (из них 6 могут использоваться в качестве ШИМ-выходов)
Аналоговые входы	6
Максимальный ток одного вывода	40 мА
Максимальный выходной ток вывода 3.3V	50 мА
Flash-память	32 КБ (ATmega328) из которых 0.5 КБ используются загрузчиком
SRAM	2 КБ (ATmega328)
EEPROM	1 КБ (ATmega328)
Тактовая частота	16 МГц

Рис.9 Характеристика ArduinoUno

Живлення

Arduino Uno може бути під’єднаний за допомогою USB-кабелю або зовнішнього джерела живлення, який вибирається автоматично. Будучи зовнішнім джерелом живлення (не USB-з’єднанням) може бути використаним як мережевий AC/DC-адаптер або батарея\акумулятор. Штекер повинен бути вставлений у відповідний під нього роз'єм живлення на платі. У випадку живлення від батареї\акумулятора, її дроти повинні бути під'єднані до виходів Vin роз'єму POWER та Gnd.

Коливання напруги зовнішнього джерела живлення повинно бути в межах від 6 до 20 В. Зменшення напруги нижче 7В може привести до некоректної роботи пристрою. Якщо напруга перевищує 12В, це може спричинити перегрів стабілізатора напруги та виведення плати з робочого стану. Для корректної роботи пристрою рекомендується використовувати джерело живлення з напругою в межах від 7В до 12В.

Список виходів живлення, які розміщені на платі:

- **VIN**. Напруга, яка надходить до Arduino від зовнішнього джерела живлення.
- **5V**. 5В напруга, яка надходить на вихід, від стабілізатора напруги на платі.
- **3V3**. 3.3В напруга, яка надходять безпосередньо від стабілізатора напруги на платі.

- **GND**. Виходи заземлення.
- **IOREF**. Надає платам розширення робочу напругу мікроконтролера Ардуїно.

Пам'ять

Об'єм пам'яті мікроконтролера ATmega328 становить 32.

Мікроконтролер також має додаткові 2 КБ пам'яті SRAM і 1 КБ EEPROM.

Входи та виходи

За допомогою функцій digitalWrite() і digitalRead(),pinMode() кожен із 14 цифрових виходів може бути використаним як вихід або вход. Ліміт напруги на виходах є 5В.Кожен вихід обмежений максимальним струмом в 40 мА. Всі виходи пов'язані з внутрішніми резисторами (за замовчуванням відключеними) номіналом 20-50 кОм. Деякі виходи Ардуїно володіють додатковим функціоналом, таким як:

- **Послідовний інтерфейс: виходи 0 (RX) та 1 (TX) –** отримують, передають дані**.**
- **Зовнішні переривання: виходи 2 і 3 –** виступають в ролі джерел переривань, при не стабільному рівні сигналу на виході**.**
- **ШІМ: виходи 3, 5, 6, 9, 10 та 11 –** повертають значення у вигляді ШІМ-сигналу.
- **Інтерфейс SPI: виходи 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK) –** здійснюється зв'язок з інтерфейсом SPI.
- **Світлодіод: 13**. При значенні HIGH-вмикається, при LOW - вимикається.
- **TWI**: виходи A4 або SDA та виходи A5 або SCL, ці виходи зв'язуються по інтерфейсу TWI.
- **AREF**. Напруга аналогових входів. Може бути використаною функцією analogReference().
- **Reset**. Призводить до перезавантаження мікроконтролера.

Зв'язок

Arduino має широку варіативність для створення з'єднання з комп'ютером, іншим Ардуїном і навіть іншими мікроконтролерами. ATmega328 – це

приймач, за допомогою якого здійснюється послідовний зв'язок завдяки цифрових виходів 0 (RX) і 1 (TX). За допомогою мікроконтролера ATmega16U2 на платі забезпечується зв'язок між приймачем та USB-портом комп'ютера і при вдалому підключенні до ПК дозволяє Ардуїно розпізнатися як віртуальний COM-порт. Для мікросхеми 16U2 не потрібно встановлювати зовнішні драйвери, тому що вона використовує стандартні драйвери USB-COM.На платформі Windows потрібен лише відповідний .inf-файл. Програмне забезпечення Ардуїно включає спеціальну програму, за допомогою якої можна зчитувати та надсилати на Ардуїно прості дані.

Під час передачі даних за допомогою мікросхеми USB-UART, коли встановлено USB-з'єднання з комп'ютером, мигатимуть світлодіоди RX та TX на платі.

RGB- світлодіод (Рис.10) – пристрій, який показує комбінацію Червоно\Зеленого\Синього кольорів.

Рис.10**RGB- світлодіод**

Резистор(Рис.11) – пристрій, який опирає потоку електрики.

Рис. 11 Резистор

Перемичка(Рис.12) – кабель-перехідник, який з’єднує Arduinoз прототипною платою.

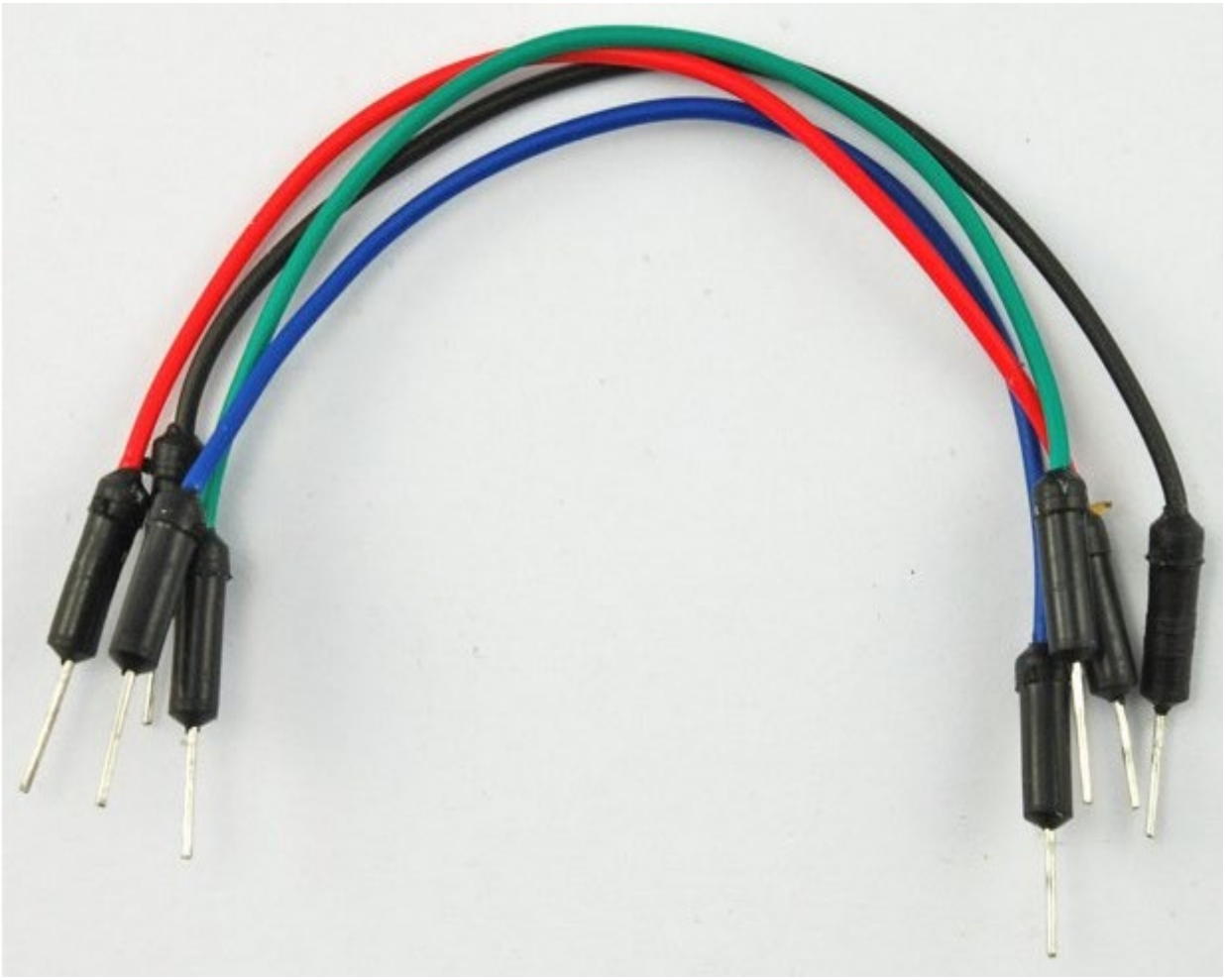

Рис.12 Перемичка

Прототипна плата (Рис.13) - плата розширення для створення власних прототипів методом паяння.

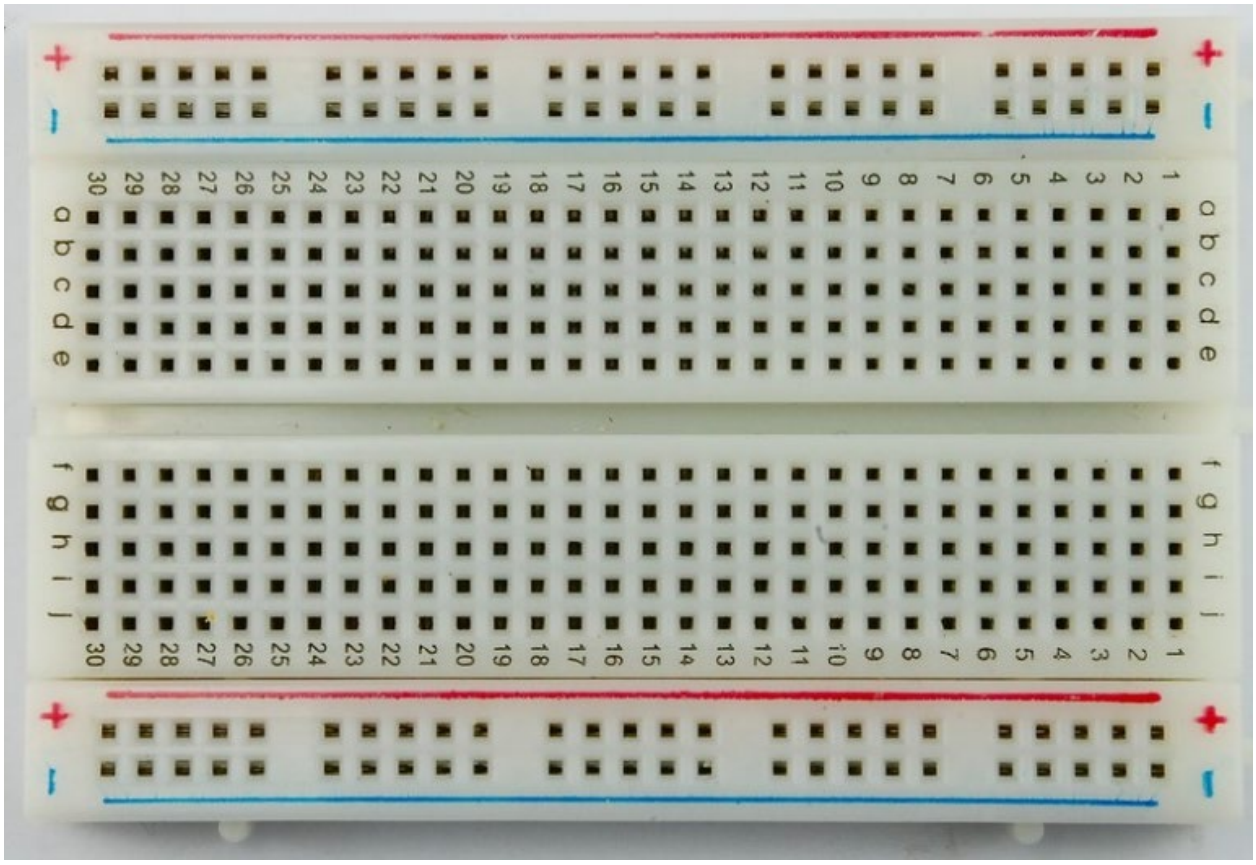

Рис.13 Резистор

Висновки до розділу 2

У дипломній роботі показано як можна керувати мікроконтролером Arduino для якого необхідно мінімальна кількість матеріалів.

Під час виконання огляду мікроконтролера Arduino, було розгялнуто його основні характеристики. Було вказано рекомендації, для коректної роботи пристрою. Показано, як вонинні підключатися компоненти для того, щоб скласти мікроконтроллер Arduino, який можна буде підключити до комп’ютера через USB-кабель.

РОЗДІЛ 3. ПРИКЛАДИ ВИКОРИСТАННЯ БІБЛІОТЕК ДЛЯ КОНТРОЛЮВАННЯ РОБОТА HAARDUINO

3.1Приклад використання бібліотеки Serialport

Для того, щоб використати Serialport, його потрібно встановити локально за допомогою команди `**npminstallserialport**`

Запис в послідовний порт виглядає дуже примітивно і просто(Рис.14).

```
void setup() {
Serial.begin(9600);
}
void loop() {
Serial.println("Hello world!");
delay(1000);
}
```

Рис. 14 Запис в Arduino через послідовний порт

Список речей, яких потрібно притримуватися:

- На ком’пютері потрібно буде знайти назву послідовного порта. Найкращий спосіб зробити це через ArduinoIDE (Інтегроване Середовище Розробки Ардуіно), Інструменти > Меню портів.
- Потрібно бути впевненим, що baudRate (швидкість передачі в бодах) співпадає з тею, що на Arduino(налаштовується в функціях налаштування)

Один з найзручніших способів використання serialport в комбінації з `@serialport/parser-readline`. Цей пакет використовується для розділення повідомлень за допомогою delimiter (надсилає дані кожного разу, як приходить нова послідовність байтів), який був вказаний.

```
const SerialPort = require('serialport');
const Readline = require('@serialport/parser-readline');

const port = new SerialPort('/dev/ttyACM0', { baudRate: 9600 });
const parser = port.pipe(new Readline({ delimiter: '\n' }));

// Read the port data
port.on("open", () => {
  console.log('serial port open');
});

parser.on('data', data =>{
  console.log('got word from arduino:', data);
});
```

Рис.15 Приклад коду з використанням Serialport

Ось простий приклад коду(Рис.15), за допомогою якого при підключенні до послідовного порту в консолі буде показано `**'serialportopened** `, а потім виведеться повідомлення `**gotmessagefromarduino: serialportopened** `.

SerialPort(path, options);

path – системний шлях до послідовного порту, який потрібно відкрити.

options – набір додаткових налаштувань, в даному випадку вказано швидкість передачі в бодах.

pipe – функція, яка передає по каналу даних додаткові функції.

Для того, щоб світлодіод мигав з затримкою 1 секунду, потрібно написати наступний код (Рис.16) :

```
// LED on pin 12
int led = 12;

// Incoming serial data
int data=0;

void setup() {
  // Pin 12 set to OUTPUT
  pinMode(led, OUTPUT);

  // Start listening on the serialport
  Serial.begin(9600);
}

void loop() {

  if(Serial.available()>0){

      // Read from serialport
      data = Serial.read();

      // Check and see if data received == 4
      if(data=='4') {
        // Blink the LED 3 times
        for(int i=0;i<3;i++){
            digitalWrite(led, HIGH);
            delay(1000);
            digitalWrite(led,LOW);
            delay(1000);
        }

        // Reset data to 0
        data=0;
     }
  }
}
```

Рис.16 Мигання світлодіоду з затримкою 1с

3.2Приклад використання бібліотеки Babbler_h

Для того, щоб почати працювати з даною бібліотекою, потрібно її завантажити за допомогою команди `**npminstallbabbler-js**` і встановити всі необхідні залежності, за допомогою команди `**npminstall**`.

Потрібно створити файл **babbler.js**, та написати наступний код(Рис.17):

```
const BabblerDevice = require('babbler-js');
const babbler = new BabblerDevice();

babbler.on('connected', function() {
    console.log("connected");

    console.log("send cmd: ping");
    babbler.sendCmd("ping", [],
        // onReply
        function(cmd, params, reply) {
            console.log("got reply on '" + cmd + " " + params + "': " + reply);
        },
        // onError
        function(cmd, params, err) {
            console.log("fail with '" + cmd + " " + params + "': " + err);
        }
    );

    console.log("send cmd: help --list");
    babbler.sendCmd("help", ["--list"],
        // onReply
        function(cmd, params, reply) {
            console.log("got reply on '" + cmd + " " + params + "': " + reply);
        },
        // onError
        function(cmd, params, err) {
            console.log("fail with '" + cmd + " " + params + "': " + err);
        }
    );
});

babbler.on('disconnected', function(error) {
    console.log("disconnected" + (error != undefined ? ": " + error : ""));
});

babbler.connect("/dev/ttyUSB1");
//babbler.connect("/dev/ttyUSB1", {baudRate: 9600});
```

Рис.17 Приклад коду з бібліотекою Bebbler_h

Де:

on(eventName, callback) – функція, яка першим аргументом приймає **eventName**і коли на робота надійде подія **eventName**буде викликано функція зворотнього виклику **callback.**

connect(pathName) – функція, яка підключається по системному шляху до послідовного порту **pathName**.

Щоб підключитися до пристрою, в консолі потрібно виконати дві прості команди: **ping** та**help –list.** Далі потрібно запустити скріпт, виконавши команду `**nodebabbler.js**`

В консолі буде відображено наступні рядки(Рис.18), символізуючи про те, що все виконалося корректно:

```
connected
send cmd: ping
send cmd: help --list
got reply on 'ping ': ok
got reply on 'help --list': help ping ledon ledoff
```

Рис. 18 Коректна робота програми

Якщо витягнути USB-кабель з роботом, програма напише останнє повідомлення і завершить свою роботу (Рис. 19):

```
disconnected: Device unplugged
```

Рис. 19 Девайс від’єднано від USB-кабелю

Далі буде цікавіший приклад (Рис.20, Рис.21), в якому буде відбуватися:

- Программа підключається до пристрою і починає вмикати\вимикати лампочку на пристрої кожні 2 секунди
- Якщо вийняти USB-кабель з роботом, програма буде намагатися перепідключатися до робота кожні 3 секунди, до того моменту поки не перепідключиться і знову почне вмикати\вимикати лампочки.

```
const BabblerDevice = require('babbler-js');

const babbler = new BabblerDevice();
let blinkIntervalId;

babbler.on('connected', function() {
    console.log("connected");
    |
    let ledstatus = "off";
    blinkIntervalId = setInterval(function() {
        if(ledstatus === "on") {
            console.log("send cmd: ledoff");
            babbler.sendCmd("ledoff", [],
                // onReply
                function(cmd, params, reply) {
                    console.log("got reply on '" + cmd + " " + params + "': " + reply);
                    ledstatus = "off";
                },
                // onError
                function(cmd, params, err) {
                    console.log("fail with '" + cmd + " " + params + "': " + err);
                }
            );
        } else { // ledstatus === "off"
            console.log("send cmd: ledon");
            babbler.sendCmd("ledon", [],
                // onReply
                function(cmd, params, reply) {
                    console.log("got reply on '" + cmd + " " + params + "': " + reply);
                    ledstatus = "on";
                },
                // onError
                function(cmd, params, err) {
                    console.log("fail with '" + cmd + " " + params + "': " + err);
                }
            );
        }
    }, 3000);
});

babbler.on('connecting', function() {
    console.log("connecting...");
});
```

Рис.20 Приклад коду з миганням світлодіоду на Babbler_h

```
babbler.on('disconnected', function(error) {
    console.log("disconnected" + (error != undefined ? ": " + error : ""));

    clearInterval(blinkIntervalId);

    setTimeout(function() {
        babbler.connect("/dev/ttyUSB0");
    }, 3000);
});

babbler.connect("/dev/ttyUSB0");
//babbler.connect("/dev/ttyUSB0", {baudRate: 9600});
```

Рис.21 Продовження Рис.20

Модернізувавши код, потрібно запустити скріпт, виконавши команду (Рис.22):

```
node babbler-basic-blink.js
```

Рис.22 Запуск babbler-basic-blink.js

Далі в консолі буде відображатися (Рис.23):

```
connecting...
connected
send cmd: ledon
got reply on 'ledon ': ok
send cmd: ledoff
got reply on 'ledoff ': ok
send cmd: ledon
got reply on 'ledon ': ok
send cmd: ledoff
got reply on 'ledoff ': ok
send cmd: ledon
got reply on 'ledon ': ok
disconnected: Device unplugged
connecting...
disconnected: Error: Error: No such file or directory, cannot open /dev/ttyUSB0
connecting...
disconnected: Error: Error: No such file or directory, cannot open /dev/ttyUSB0
connecting...
connected
send cmd: ledon
got reply on 'ledon ': ok
send cmd: ledoff
got reply on 'ledoff ': ok
send cmd: ledon
got reply on 'ledon ': ok
disconnected: Device unplugged
```

Рис. 23. Міграція світлодіодів з підєднанням\відєднанням USB-кабелю

В процесі виконання, був витягнутий USB-кабель з роботом, а потім знову вставлений.

3.3 **Приклад використання бібліотеки Johnny-five**

Для того, щоб почати працювати з даною бібліотекою, потрібно встановити її за допомогою команди (Рис.24)

```
npm install johnny-five
```

Рис. 24ІнсталяціяJohnny-five

Необхідне програмне забезпечення (Рис.25):

- 1 Arduino (Genuino) Uno
- 1 СВД
- 1 прототипна плата
- 2 дротяні перемички з двома штировими кінцями (1 червона, 1 чорна)
- 1 резистор (200 – 330 Ω)

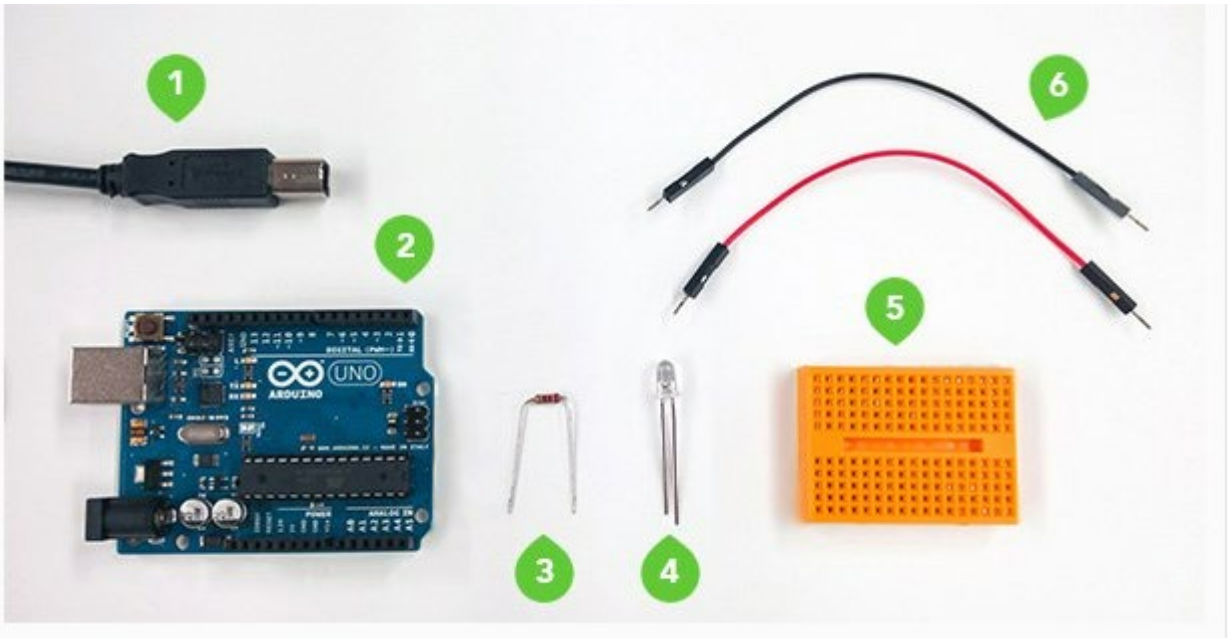

Рис. 25 Необхідне програмне забезпечення

Для того, щоб все працювало, потрібно зєднати всі компоненти, як на (Рис.26)

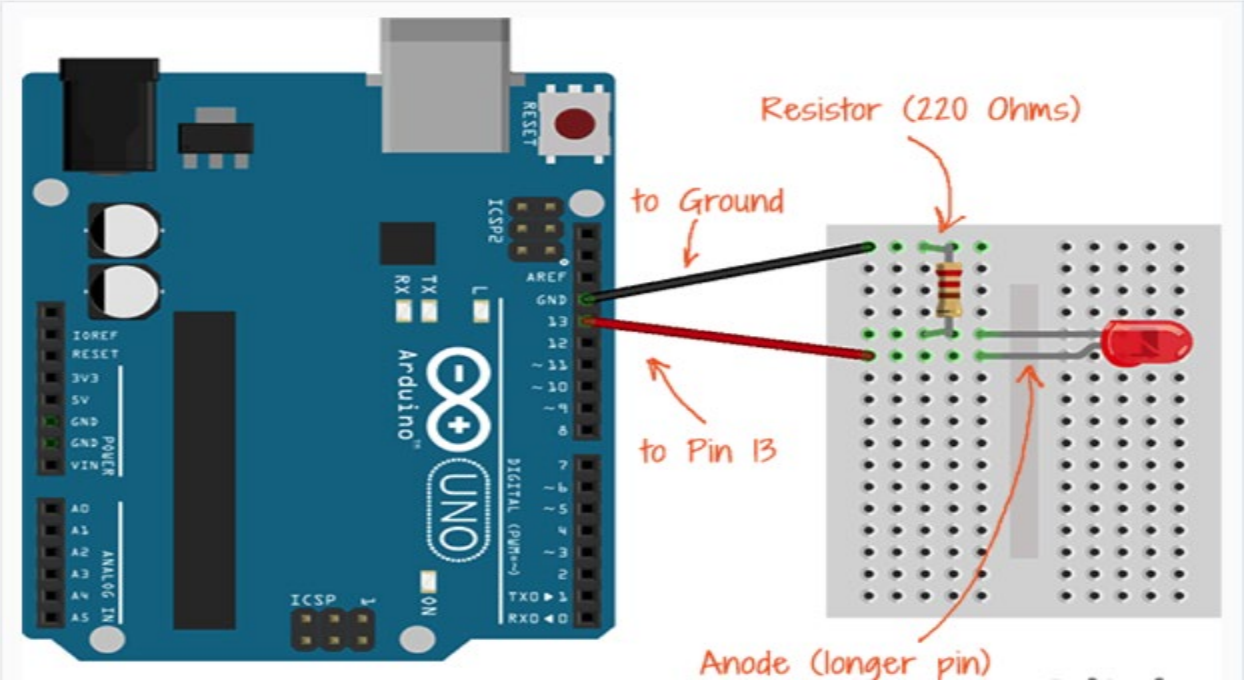

Рис.26 Складений Arduino

Далі потрібно створити файл **Johnny-five.js**і написати наступний код(Рис.27):

```
const five = require('johnny-five');
const board = new five.Board();

board.on('ready', function() {
  const led = new five.Led(13);
  led.blink(500);
});
```

Рис. 27. Код, для загорання світлодіоду на Johnny-five

Де:

on(eventName, callback) – функція, яка першим аргументом приймає **eventName**і коли на робота надійде подія **eventName**буде викликано функція зворотнього виклику **callback.**

five.Led(pin) – функція, яка підключається до 13 порту.

blink(interval) – функція, яка буде кліпати з інтервалом у **interval** мілісекунд. Для того, щоб виконати даний код, потрібно запустити команду (Рис.28)

```
node Johnny-five.js
```

Рис.28 Запуск Johnny-five

Світлодіод повинен кліпати з інтервалом у 500 мілісекунд (Рис 29).

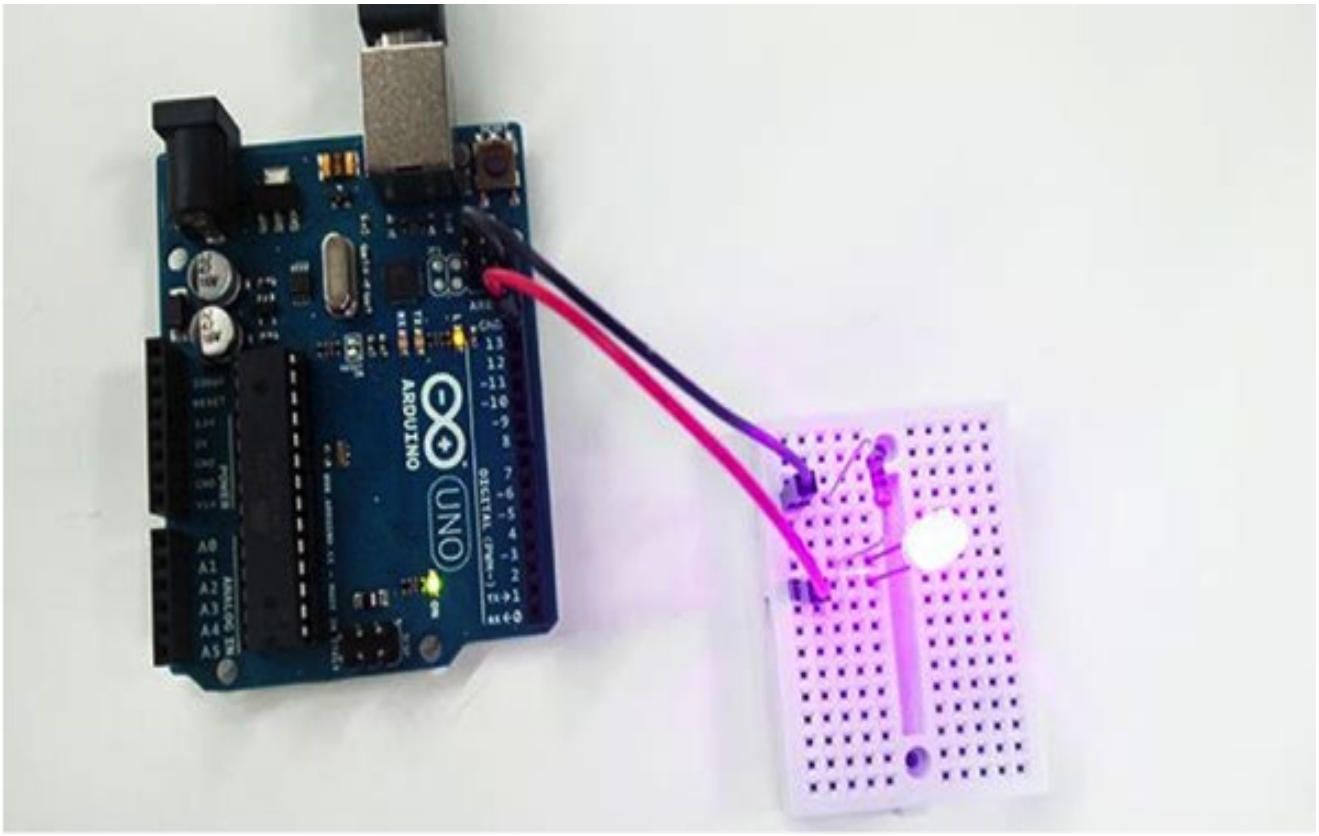

Рис. 29 Мигаючий світлодіод

Також можна створити веб інтерфейст для зміни СВД за домопомогою повзунка. Для цього потрібно використати HTMLтег <inputid="green"type="range"min="0"max="255"step="1"value="0"/>, який дозволить намалювати повзунок-перемикач, завдяки якому користувач зможе змінювати значення ‘R’, ‘G’, ‘B’. Значення кожного кольору знаходиться в діапазоні від 0 до 255, крок зміни величини 1.

За допомогою цього повзунка відбувається контроль інтенсивності зеленого кольору.

Для того, щоб з’єднати UI(UserInterface – те що бачить користувач на сайті), потрібно використати бібліотеку PubNub(Рис. 30), для цього потрібно її встановити командою (Рис. 31):

```
npm i pubnub
```

Рис. 31ВстановленняPubnub

Також, для того, щоб користуватися бібліотекою, потрібно згенерувати власний API (Рис. 32) (Application Programming Interface – програмний інтерфейс програми) на офіційному сайті Pubnub.

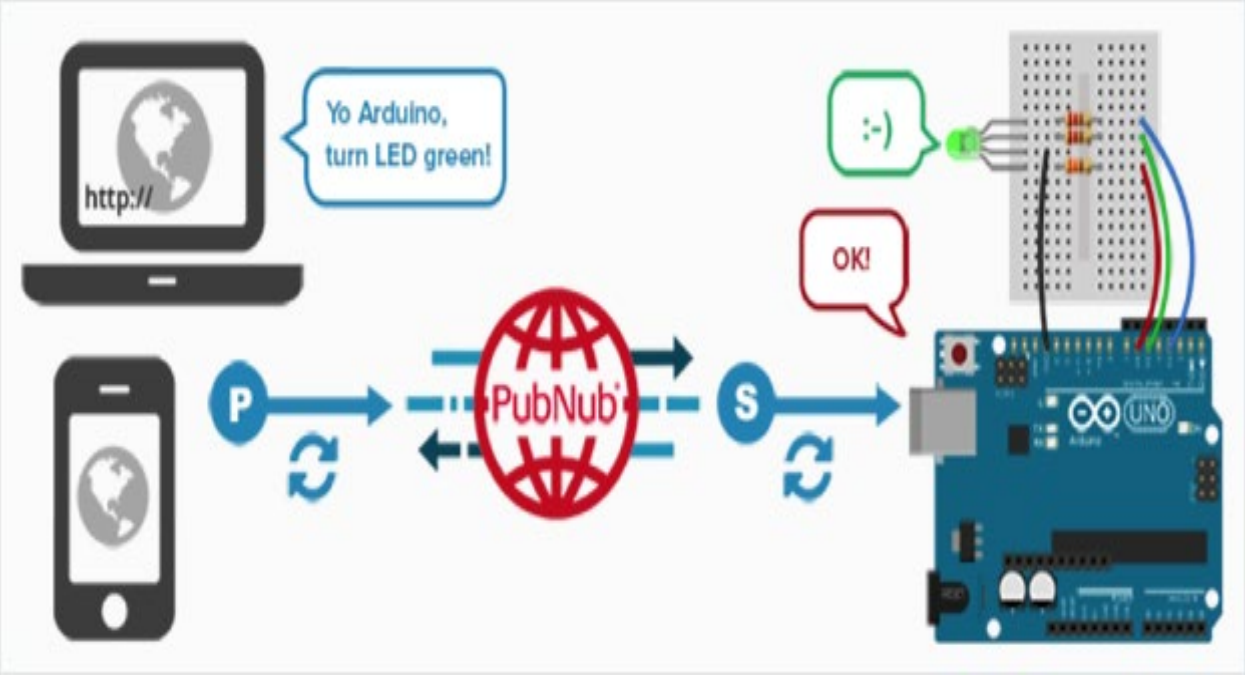

Рис.30 Мережа потоку даних Pubnub

```
const pubnub = require('pubnub').init({
    publish_key: 'pub-c-0b43969b-341d-41f5-a85e-0bd9d30404b8',
    subscribe_key: 'sub-c-cb24903e-c9f4-11e5-b684-02ee2ddab7fe'
});
```

Рис.32 Згенерований APIдля Pubnub

Необхідне програмне забезпечення(Рис. 33):

- Arduino (Genuino) Uno
- 1 RGB-(* система передавання кольору RGB (Red-Green-Blue)) СВД (загальний катод)
- 1 прототипна плата
- 4 дротяні перемички з двома штировими кінцями (червона, чорна, зелена, синя)
- 1 резистор (220 Ω x 2, 330 Ω x 1)

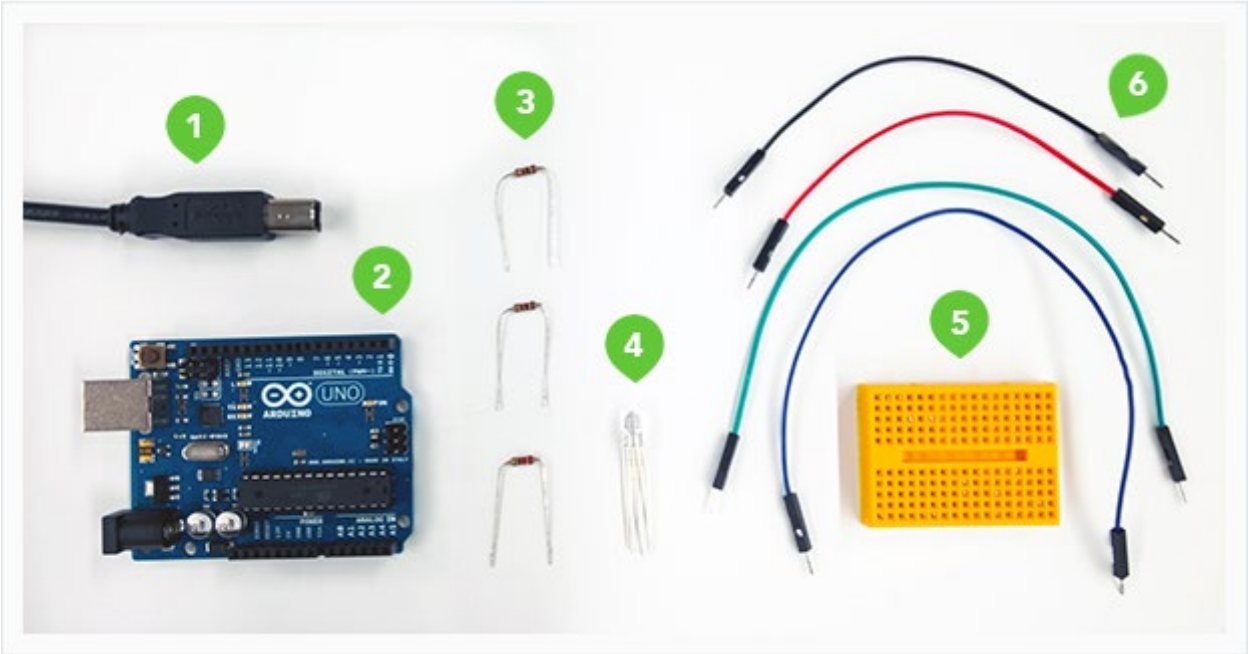

Рис.33. Програмне забезпечення для Arduino з використанням бібліотеки Pubnub

На Рис. 34 показано, як повинен виглядати зібраний Arduino

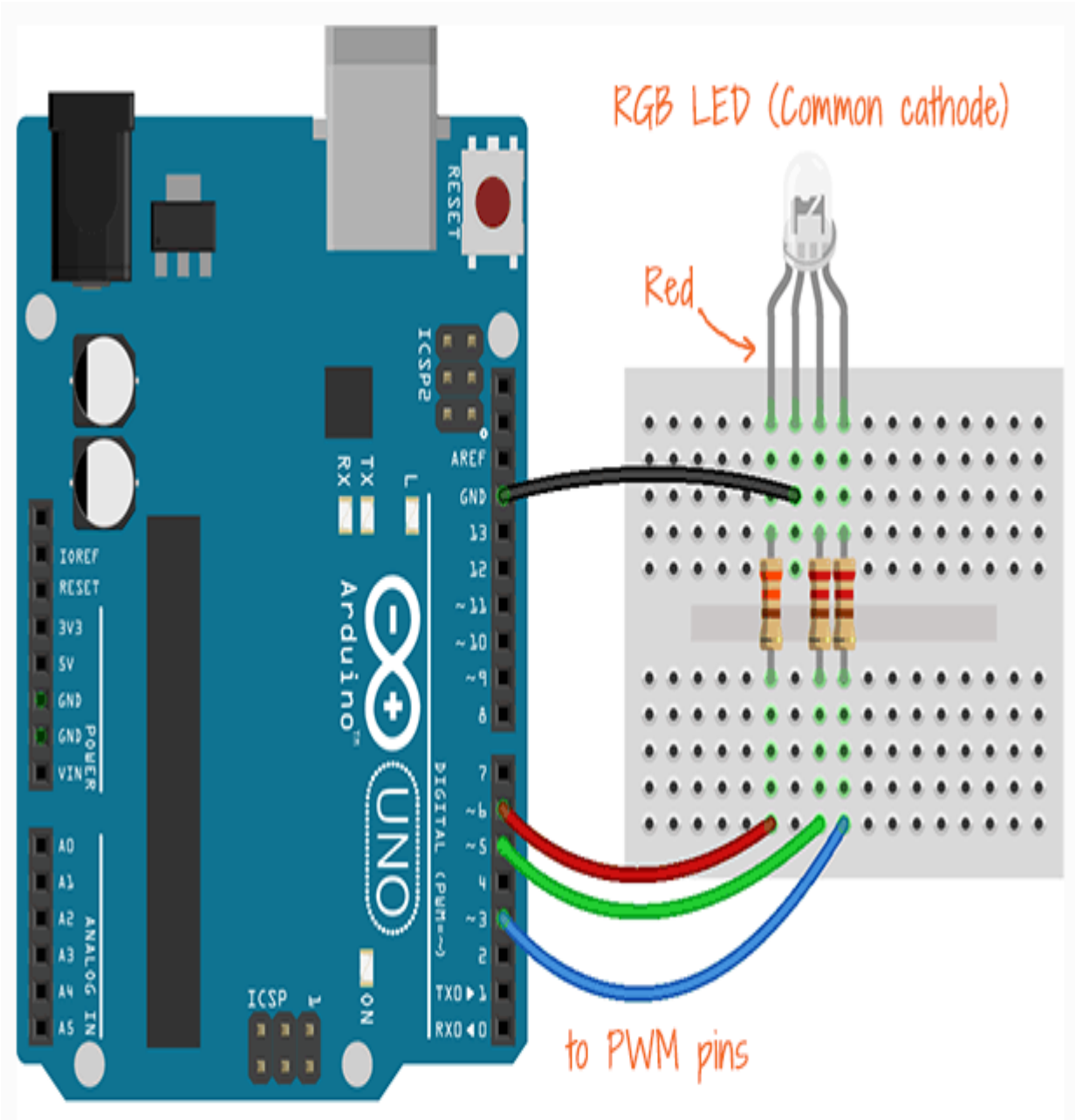

Рис. 34. Зібране програмне забезпечення для Arduino з використанням бібліотеки Pubnub

Для того, запустити програму, потрібно створити index.htmlфайл (Рис.35), в якому будуть створені повзунки за допомогою тегу <input/>

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>

    <div>
        <div class="caption red">Red</div>
        <label for="red" class="brightness fa">
            <input id="red" type="range" min="0" max="255" step="1" value="0">
        </label>
    </div>

    <div>
        <div class="caption green">Green</div>
        <label for="green" class="brightness fa">
            <input id="green" type="range" min="0" max="255" step="1" value="0">
        </label>
    </div>

    <div>
        <div class="caption blue">Blue</div>
        <label for="blue" class="brightness fa">
            <input id="blue" type="range" min="0" max="255" step="1" value="0">
        </label>
    </div>

</body>
<script src="./Johnny-five.js"></script>
</html>
```

Рис.35index.htmlфайл з повзунками

```
const five = require('johnny-five');
const red = document.getElementById('red');
const green = document.getElementById('green');
const blue = document.getElementById('blue');
const brightness = {r: 0, g: 0, b: 0};
```

Рис.36 Код, для підключення до UI(UserInterface) за допомогою Pubnub

```
five.Board().on('ready', function() {
  console.log('ready');

  // Initialize the RGB LED
  var led = new five.Led.RGB({
    pins: {
      red: 6,
      green: 5,
      blue: 3
    }
  });

  led.on();
  led.color({red: 0, blue: 0, green: 0});

  const pubnub = require('pubnub').init({
    publish_key: 'publish_key',
    subscribe_key: 'subscribe_key'
  });

  const channel = 'smart-led';

  pubnub.subscribe({
    channel: channel,
    callback: setLedColor,
    connect: initLedColor,
    error: function(err) {console.log(err);}
  });

  function setLedColor(m) {
    led.color({red: m.r, blue: m.b, green: m.g});
    console.log( 'color change to...' );
    console.log( led.color() );
  }

  function initLedColor() {
    pubnub.history({
      channel: channel,
      count: 1,
      callback: function(messages) {
        messages[0].forEach(function(m) {
          setLedColor(m);
        });
      }
    });
  }
});
```

Продовження Рис.36

```
function publishUpdate(data) {
    pubnub.publish({
        channel: 'smart-led',
        message: data
    });
}

pubnub.subscribe({
    channel: channel,
    message: resetSliders,
    connect: initSliders,
});

function resetSliders(m) {
    red.value = brightness.r = m.r;
    green.value = brightness.g = m.g;
    blue.value = brightness.b = m.b;
}

function initSliders() {
    pubnub.history({
      channel: channel,
      count: 1,
      callback: function(messages) {
        messages[0].forEach(function(m) {
          console.log(m);
          resetSliders(m);
        });
      }
    });
}
```

Продовження Рис.36

```
red.addEventListener('change', function(e){
    brightness.r = this.value;
    publishUpdate(brightness);
}, false);

green.addEventListener('change', function(e){
    brightness.g = this.value;
    publishUpdate(brightness);
}, false);

blue.addEventListener('change', function(e){
    brightness.b = this.value;
    publishUpdate(brightness);
}, false);
```

Продовження Рис. 36

Далі, запустивши код через сервер, при зміні повзунків у браузері, світлодіоди на Arduino будуть змінювати свій відтінок.

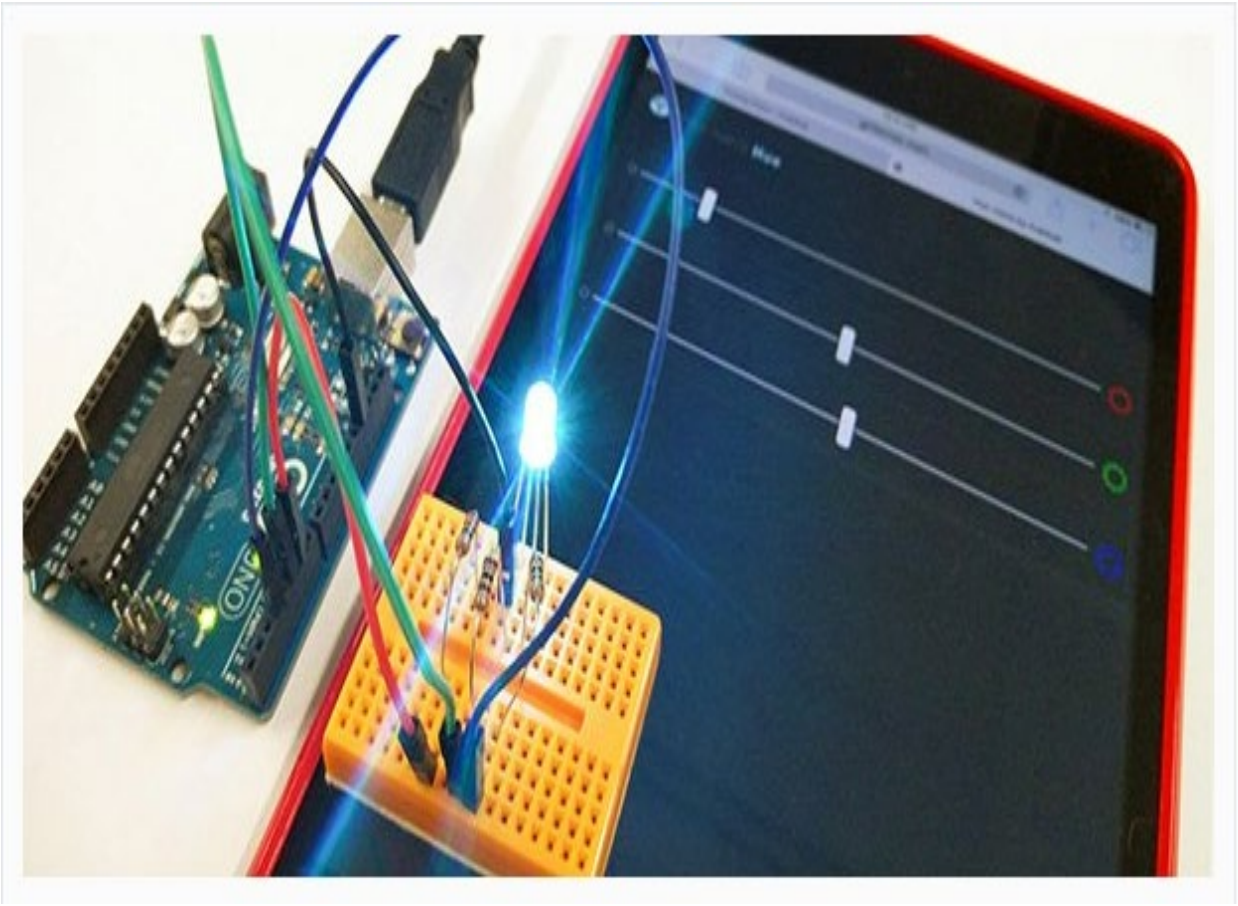

Рис. 37 При зміні повзунка на екрані, змінюється колір світлодіода

Висновки до розділу 3

Для кожної бібліотеки, яка може керувати роботом на Arduino, було написано програму і показано як за допомогою функцій та методів бібліотеки можна керувати роботом. Було наведено приклади коду, в яких за допомогою консолі можна побачити, як поводив себе робот на Arduino при підключенні, вимкненні, перепідкюченні. Було прикріплено скріншоти на яких видно, як загорається світлодіод.

На мою думку, найкраще себе показала бібліотека Johnny-five, тому що вона проста у використанні, має досить велику аудиторію, яка постійно покращує якість коду і робить якнайпростіше керування робота на Arduino. Ця бібліотека також довзоляє працювати не тільки з мікроконтролером Arduino, але й з рядом інших мікроконтролерів.

ВИСНОВКИ

У дипломній роботі було вибрано найкращуJavaScript бібліотеку для контролювання робота, якості якого було використано мікроконтролер Arduino, для якого потрібно мінімальна кількість матеріалів, щоб його зібрати. За допомогою бібліотеки Johnny-five можна створювати як примітивні контролери, які вміють вмикати\вимикати світлодіод, так і більше конструйовані системи, такі як контролер, який буде слідкувати за температурою в кімнаті за допомогою датчиків відкривати вікно, коли температура висока, або закривати його, коли температура опустилася.

Я вважаю, що бібліотека Johnny-five показала себе найкраще, тому що вона дуже просто у використанні, у неї велика спількота, яка розвиває даний проєкт і вона кросс-платформенна, тобто вона може пікдлючатися до більшості відомих нині контролерів і працювати з ними.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. https://v8.dev/
2. 'https://nodejs.dev/learn/the-v8-javascript-engine
3. https://nodejs.org/uk/
4. https://learn.javascript.ru/
5. https://developer.mozilla.org/ru/docs/Learn/Server-side/Express_Nodejs/development_environment
6. 'https://www.w3schools.com/nodejs/
7. https://doc.arduino.ua/ru/hardware/Uno
8. https://arduino.ua/index.php?categoryID=40&show_all=yes
9. https://code.tutsplus.com/uk/tutorials/how-to-create-a-smart-device-with-arduino-and-nodejs-using-pubnub--cms-25508
10. https://github.com/girliemac/arduino-led-rgb/blob
11. https://roboticsbackend.com/control-arduino-with-javascript-and-johnny-five-firmata-protocol/
12. https://github.com
13. https://circuitdigest.com/microcontroller-projects/arduino-nodejs-tutorial-control-led-brightness-with-web-interface
14. https://desertbot.io/blog/control-arduino-with-nodejs
15. https://flaviocopes.com/johnny-five/
16. http://johnny-five.io/
17. https://github.com/pubnub/javascript
18. https://www.npmjs.com/package
19. https://medium.com/@machadogj/arduino-and-node-js-via-serial-port-bcf9691fab6a
20. https://serialport.io/docs/api-serialport#parsers
21. https://www.ladyada.net/learn/arduino/lesson4.html
22. https://habr.com/ru/post/316194/

23. https://habr.com
24. https://habr.com/ru/post/315480
25. https://www.jscamp.app/ru/docs/javascript25/
26. http://developer.alexanderklimov.ru/arduino/docs/spec.php
27. https://stackoverflow.com/questions/38501650/led-blink-with-serial-port